Report No: 01
Report Name: Midpoint Circle Drawing Algorithm

Procedure:
Given:
Center point of circle = (x_0, y_0)
Radius of circle = R

Step 1:
Assign the starting point coordinates (x_0, y_0) as:
$x_0 = 0, y_0 = R$
Step 2:
Calculate the value of the initial decision parameter:
$P_0 = 1 - R$

Step 3:
The current point is (x_k, y_k), and the next point is (x_{k+1}, y_{k+1}).
Here, two cases exist:

Case 1:
If $P_k < 0$:
$x_{k+1} = x_k + 1$
$y_{k+1} = y_k$
$P_{k+1} = P_k + 2x_{k+1} + 1$

Case 2:
If $P_k \geq 0$:
$x_{k+1} = x_k + 1$
$y_{k+1} = y_k - 1$
$P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}$

Step 4:
If the given center point (x_0, y_0) is not $(0, 0)$, then adjust and plot the points as:
$x_{\text{plot}} = x + x_0$

$y_{\text{plot}} = y + y_0$
Step 5:
Keep repeating steps 3 and 4 until $x_{\text{plot}} \geq y_{\text{plot}}$.

Step 6:
Step 5 generates all points for one octant. Using symmetry, plot points for all eight octan

Report No: 02
Report Name: Bresenham Circle Drawing Algorithm

Procedure:
Given the center point of the circle = (x_0, y_0)
Radius of the circle = R

Step 1: Assign the starting point coordinates (x_0, y_0) as:
- $x_0 = 0, y_0 = R$

Step 2: Calculate the value of the initial decision parameter p_0 as:
- $p_0 = 3 - 2 \times R$

Step 3: The current point is $(x_{\square}, y_{\square})$, and the next point is $(x_{\square+1}, y_{\square+1})$. Decision parameter $p_{\square}$:
Follow two cases:
- *Case 1:* If $p_{\square} < 0$
 $x_{\square+1} = x_{\square} + 1$
 $y_{\square+1} = y_{\square}$
 $p_{\square+1} = p_{\square} + 4 \times x_{\square+1} + 6$

- *Case 2:* If $p_{\square} >= 0$
 $x_{\square+1} = x_{\square} + 1$
 $y_{\square+1} = y_{\square} - 1$
 $p_{\square+1} = p_{\square} + 4 \times (x_{\square+1} - y_{\square+1}) + 10$

Step 4: If the given input point (x_0, y_0) is not (0, 0), then do the following and plot the points:
- $x_{\square\square o\square} = x_e + x_0$
- $y_{\square\square o\square} = y_e + y_0$

Where x_e, y_e are the current values of $x_{\square}$, $y_{\square}$.
Step 5: Keep repeating *Step 3* and *Step 4* until $x_{\square\square o\square} \neq y_{\square\square o\square}$.

Step 6: *Step 5* generates all the points for one octant.

Report No: 03
Report Name: 2D Translation

Procedure:
Let initial coordinates of the object = $(x_{o}\square_{x}, y_{o}\square_{o})$

New coordinates of the object after translation = $(x\square_{ev}, y\square_{ev})$
Translation vector = (T_{x}, T_{γ})

Given a translation vector (T_{x}, T_{γ}):
- $x\square_{ev} = x_{o}\square_{o} + T_{x}$
- $y\square_{ev} = y_{o}\square_{o} + T_{\gamma}$

Code:
```
rectangle(p[0][0], p[0][1], p[1][0], p[1][1]);
p[0][0] = p[0][0] + T[0];
p[0][1] = p[0][1] + T[1];
p[1][0] = p[1][0] + T[0];
p[1][1] = p[1][1] + T[1];
rectangle(p[0][0], p[0][1], p[1][0], p[1][1]);
```

Report No: 04
Report Name: Scaling in 2D Translation

Procedure:
Let initial coordinates of the object = $(x_{o}\square_{o}, y_{o}\square_{o})$

Scaling factor for x-axis = S_x
Scaling factor for y-axis = S_γ

New coordinates of the object after scaling = $(x\square_{ev}, y\square_{ev})$

This scaling is achieved by:
- $x\square_{ev} = x_{o}\square_{o} \times S_x$
- $y\square_{ev} = y_{o}\square_{o} \times S_\gamma$

Code:
```
void Scale(float x, float y, float Sx, float Sγ) {
   float x□ev = x * Sx;
   float y□ev = y * Sγ;
   Scale(x, y, Sx, Sγ);
}
```